



Functional modelling of complex multi-disciplinary systems using the enhanced sequence diagram

Unal Yildirim¹ · Felician Campean¹

Received: 2 November 2019 / Revised: 22 June 2020 / Accepted: 27 June 2020
© The Author(s) 2020

Abstract

This paper introduces an Enhanced Sequence Diagram (ESD) as the basis for a structured framework for the functional analysis of complex multidisciplinary systems. The ESD extends the conventional sequence diagrams (SD) by introducing a rigorous functional flow-based modelling schemata to provide an enhanced basis for model-based functional requirements and architecture analysis in the early systems design stages. The proposed ESD heuristics include the representation of transactional and transformative functions required to deliver the use case sequence, and fork and join nodes to facilitate analysis of combining and bifurcating operations on flows. A case study of a personal mobility device is used to illustrate the deployment of the ESD methodology in relation to three common product development scenarios: (i) reverse engineering, (ii) the introduction of a specific technology to an existent system; and (iii) the introduction of a new feature as user-centric innovation for an existing system, at a logical design level, without reference to any solution. The case study analysis provides further insights into the effectiveness of the ESD to support function modelling and functional requirements capture, and architecture development. The significance of this paper is that it establishes a rigorous ESD-based functional analysis methodology to guide the practitioner with its deployment, facilitating its impact to both the engineering design and systems engineering communities, as well as the design practice in the industry.

Keywords Sequence diagrams · Functional requirements · Function architecture · Structure architecture · Complex multidisciplinary systems

1 Introduction

Meeting market and consumer demands in a rapidly changing technological environment places significant technical challenges on product design and development companies. Engineered systems are commonly based on a mix of electro-mechanical structural components with electronics, controls and software, increasingly connected with the broader environment. For example, the term cybertronic systems have been coined to define systems which interconnect computing and mechatronics (Eigner et al. 2014). From a product design and development viewpoint, this reflects the organisational evolution towards multidisciplinary, particularly due to the increased use of embedded computing and connectivity (Eigner et al. 2016; Campean et al. 2019).

Many researchers have discussed that increased multidisciplinary introduces important integration challenges to identify interactions between systems from different engineering disciplines early in the engineering design process (Tomiya et al. 2007; Van Beek and Tomiya 2009; Lindemann et al. 2009 and D'Amelio et al. 2011; Hehenberger et al. 2016). As noted by Karrer-Müller et al. (2016), Karkare (2018), Burcicki (2019) and Albers et al. (2019), these challenges also increase the importance of system architecture development to address evolving functional and structural requirements in the improvement of an existing product and the development of new product generation through the integration of a new technology. This is particularly important for many systems currently designed and developed, where software is increasingly used to define the product functionality, as noted by Karkare (2018) with respect to autonomous vehicle development.

Historically, the so-called V-model (Forsberg and Mooz 1998) has been usefully employed as a reference model for the development of technical systems, including automotive

✉ Unal Yildirim
uyildir1@bradford.ac.uk

¹ Automotive Research Centre, Faculty of Engineering and Informatics, University of Bradford, Bradford, UK

(Graessler et al. 2018; Pfeffer et al. 2019), with the left side of the V-model being based on the hierarchical cascade of functional requirements from the top-level system to the component level. However, the V-model has limitations with highly complex systems and features, as exemplified by the autonomous vehicles, where it is often difficult to provide a full specification of requirements due to complexity and operational uncertainty. The development of this type of systems, including hybrid (human–machine) autonomous systems, requires carrying out modelling activity for largely unknown scenarios, for which models are not available. In general, the introduction of technologically advanced features on existing systems is recognised as a methodological challenge across industry domains (Campean et al. 2020). Alternative incremental approaches to system development and requirements capture that can accommodate the identification of changing or new requirements as the system is developed have been increasingly considered (Pfeffer et al. 2019).

From a product development process viewpoint, these challenges amplify the need for robust function modelling frameworks, to support common system design and analysis activities such as:

- (a) generic functional analysis early in the product creation process to develop the model basis for the requirements analysis, architecture design and integration;
- (b) modelling the introduction of a new technology onto an existent system, integrating new functionality with the carry-over system functional and architectural model;
- (c) modelling the introduction of new user-focussed control features to an existing system architecture.

In industrial practice functional reasoning revolves around modelling the flows through the system, generally underpinned by the Pahl and Beitz (2007) framework, which suggest that all technical systems involve technical processes whose functional model can be represented in terms of the flows of material, energy and information/signal. As noted by Eisenbart (2014), Pahl and Beitz's flow-based thinking has been widely adopted in mechanical engineering, electrical engineering, mechatronic system development and Product Service System design literature. It also provides the basis for many of the well-established modelling schemes, including Stone and Wood (2000), Ulrich and Eppinger (2003) and Ullman (2010). Recently, Short and Van Bossuyt (2015), Markos and Dentsoras (2015), Liu et al. (2015), Yildirim et al. (2017) and Zurita et al. (2018) exemplified different approaches to flow-based thinking in function modelling.

However, function modelling for early concept analysis requires focus centred on understanding the stakeholder goals as functional outcomes, and the generic mapping of

the process involving a chain of user and device-centric activities to enable the delivery of functionality that fulfils the goals. Modelling such scenarios directly in a flow-based paradigm is still a challenge for practitioners with the current modelling methodologies.

Model-Based Systems Engineering (MBSE) methodology has evolved to support requirements identification and management along with the design, analysis, verification and validation of complex systems (Micouin 2014; Cameron 2018; Brusa et al. 2018; Apostolov et al. 2018; Borky and Bradley 2019; Ferrogali et al. 2019; Tang et al. 2019). System Modelling Language (SysML) (OMG 2019) emerged as a prevalent MBSE tool for model-based development of multidisciplinary systems (Albers and Zingel 2013; Barbieri et al. 2014). Its strength stems from the fact that it enables designers to create and manage primarily graphical models of the engineering system via well-established diagrams. Behaviour diagrams of SysML provide a means of capturing functional requirements from the operational concept of a system across its lifecycle use cases. The realization of use cases is described using sequence diagrams, activity diagrams and state machine diagrams. However, while the MBSE modelling environment provides an effective way of modelling functional related activities early in the system design analysis, it does not provide a robust methodology, methods and tools support for the rigorous flow-based functional modelling, which is essential for the integration analysis of systems that have a significant physical behaviour.

Observations of industry practice by the authors, in the context of collaborative research with an automotive OEM, further substantiated these difficulties and provided motivation for this research. The authors noted that the MBSE approach used to capture the functional logic of a proposed SoI concept based on the envisaged use case scenarios (e.g. an autonomous cruise control system), rely on a variety of semi-structured methods, including function flow diagrams, activity diagrams and sequence diagrams, generally driven from a software and algorithmic data flow perspective. These methods tend to capture the functional requirements as transmission/transaction type functions, rather than transformative functions with appropriate engineering detail on attributes and metrics. This is often problematic for a variety of reasons, including the fact that it does not facilitate the evaluation of risks associated with the added complexity introduced by the new feature as a SoI within the context of the existing vehicle (as the System of Systems, SoS). Significant decisions for product development gateways require that risks are robustly evaluated before a significant amount of work is invested in the development of system design solutions, and also the validation of the logic and architecture, including integration requirements within the whole system, have been consistently and coherently performed. From an OEM

point of view, this prompts further methodological challenges in the integration of the evaluation of risks based on the existing Function Failure Mode Avoidance methodology used in the automotive industry (see for example, Campean et al. 2013) with the early model-based concept and feature analysis.

Thus, we conclude that there is a significant gap in both research and practice in relation to methods that support rigorous functional flow analysis early in the system design and analysis phase. The research presented in this paper aims to address this gap by introducing a rigorous functional flow-based modelling method and scheme, referred to as the Enhanced Sequence Diagram (ESD). The core idea of the ESD is the augmentation of the conventional sequence diagram with a systematic notation of flow-based operations, as discussed earlier by Yildirim (2015) and Campean and Yildirim (2017), based on preliminary work and examples of motivating industrial case studies. This paper reports on substantial novel work towards the development and establishment of the ESD notation and methodology. The key aspects covered in this paper include:

1. a coherent argument underpinning the establishment of a rigorous and complete ESD notation;
2. a functional modelling methodology based on the ESD, including a set of process heuristics; and
3. a set of validation case studies, offering a comprehensive illustration of the deployment of the ESD methodology within three product development scenarios associated with a personal mobility system (a hybrid electric bicycle), i.e. (i) reverse engineering; (ii) introduction of a new technology on an existing system; and (iii) introduction of new user-centric innovation control feature to an existing system architecture.

The proposed ESD method not only bridges the gap between the MBSE methods and tools and the well-established and proven rigorous methods for functional modelling in engineering design, but also, as argued in this paper, offers a powerful function modelling method for use-case based analysis and architecture synthesis support for complex systems.

The organisation of the paper is as follows: Sect. 2 reviews SysML behaviour modelling diagrams, with a strong focus on sequence diagrams. Section 3 presents the ESD, including definitions and conventions for the representation of the elements of the diagram covering all types of operations of flows. Section 4 introduces the application of the ESD to an electric hybrid bicycle. Section 5 discusses key aspects of the ESD-based functional modelling with reference to the case study, while Sect. 6 concludes on the value and impact of the ESD framework.

2 Review of related work

SysML behaviour diagrams attempt to capture and represent the functionality of a system in terms of how it operates (Botham et al. 2017). This includes use case diagrams to capture graphically the uses of a system (Cockburn 2000), while state machine, sequence and activity diagrams detail functionalities represented in use case diagrams in different ways. State machine diagrams describe state-dependent behaviour of an entity in terms of its states and the transitions between them (Booch et al. 2007). Sequence diagrams capture exchange-based behaviour with reference to a sequence of messages between actors or entities, whereas activity diagrams represent flow-based behaviour of a system. SysML sequence and activity diagrams are based on similar graphical representations used in Unified Modeling Language (UML) (OMG 2017), adapted with lightweight customizations, e.g. renaming Classes as Blocks.

This section focuses on the current use of activity and sequence diagrams as this provides context for the development of the Enhanced Sequence Diagram notation.

2.1 Activity diagram

The activity diagram is akin to a traditional functional flow diagram and it can be used in the modelling of a system in terms of the flow of inputs, outputs, and controls. An action is typically represented as a rectangle with round corners. Inputs and outputs of an activity diagram are called tokens which may represent units of information, material, or energy. Tokens are placed on pins, which are object nodes for inputs and outputs to actions. Pins are generally shown as small rectangles attached to the action rectangles (OMG 2017). An action processes tokens on input pins and places on output pins for other actions to accept, i.e. output token of an action is the input token of the next action. While pins are connected using object flows, actions can be connected via control flows. The relationship between the activities of a system and its structure needs to be established as system design progresses (Friedenthal et al. 2012; Holt and Perry 2019). Botham et al. (2017) commented that Activity Diagrams do not provide enough information about the relationships between the inputs and outputs of an activity to permit strong system analysis. For example, the analysis in Fig. 1, which illustrates an activity diagram for the “control power” activity of an automobile, does not provide appropriate detail of the attributes and metrics of the tokens (e.g. gear select) for an engineering analysis.

2.2 Sequence diagrams

Sequence Diagrams (SD) originate in the Message Sequence Charts of the Specification and Description

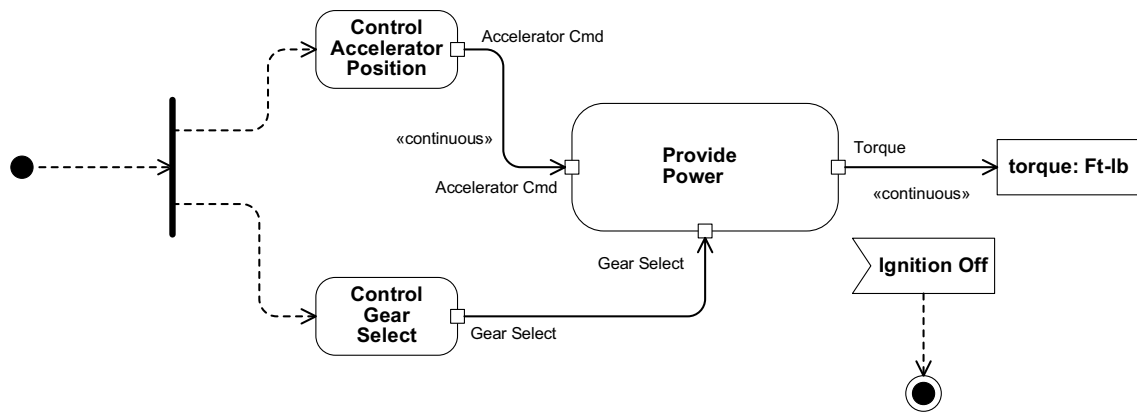


Fig. 1 Control Power Activity Diagram (adapted from Friedenthal et al. 2012)

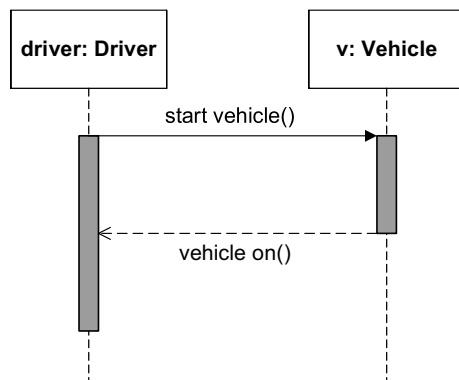


Fig. 2 “Turn On Vehicle” Sequence Diagram (adapted from Friedenthal et al. 2012)

Language (SDL) (SDL-RT 2013). UML/SysML sequence diagrams have many common features with the SDL (Weilkiens 2006; Xie et al. 2009). Human or other external entities in a SysML sequence diagram are represented by what is termed an actor. An actor or multiple actors interact with the system for the fulfilment of a system use case. Both the actor and the system are represented in rectangles with dashed lines (commonly referred to as “lifelines”) descending from the base of the rectangles with respect to time. Behaviours arising from interactions between the actor(s) and the system are represented in terms of a sequence of message exchanges between these lifelines (Friedenthal et al. 2012; Holt and Perry 2019), as illustrated in Fig. 2.

While sequence diagrams focus on the mapping of message-based information interactions between the system and the actors, Friedenthal et al. (2012) discussed that the passage of material and energy can also be indicated

through arguments of the message, opening up for the use of the SD on more general modelling of physical systems.

2.3 Critique of sequence diagrams

While the SDs are useful for the modelling of use-case scenarios, in terms of the analysis of multidisciplinary systems, conventional SD provides insufficient information to extract functional requirements associated with exchanges of material, energy and signal/information. Several researchers have attempted to address this gap by extending sequence diagrams to support function modelling of systems. For example, the synchronization-adorned UML (saUML) of Xie et al. (2009) proposes that changes in the state of an actor are captured via a message articulated in verb-noun format. Zingel et al. (2012) used sequence diagrams in the definition of functions for specific operations of a system by representing a function by an arrow and mapping functions between the lifelines of the actors and the system in terms of concrete events. Similarly, Piques (2014) complemented the sequence diagrams by functions, while Daily et al. (2016), Chen et al. (2015) and Gouhar et al. (2018) depicted functionality of a system with a sequence diagram, including the representation of system parameters. These extensions provide support for the use of sequence diagrams in function modelling, however, the question of how to use sequence diagrams for the systematic analysis of a system with multiple combinations of flows material, energy and information remains unaddressed.

In general, a complex multidisciplinary system has to deal with different combinations of input and output flows of energy, material and information during its operation, corresponding to a particular use case. There can be multiple input states and one output state, one input state and multiple output states, or multiple input and multiple output states. In SysML Activity and State Machine Diagrams, this representation challenge is addressed by introducing join

and fork nodes, represented as vertical lines. For example, the analysis in Fig. 1 (Friedenthal et al. 2012) shows that the use case scenario requires the concurrent initiation of 2 independent activities, i.e. both the Control Accelerator Position activity and the Control Gear Select activity, via a fork node. Efforts to adapt SD for such problems within UML/SysML are exemplified by the introduction of interaction operators, including operators for alternative and parallel routes (Berkenkötter 2003; Micskei and Waeselynck 2011). In spite of the improvements made, the usability of sequence diagrams for the analysis of systems with multiple operations (including combining and branching) for material, energy and information flows is still unclear. The focus is mostly on information flow as exchanges of messages, and even there it is not clear how to represent multiple incoming messages to generate an outgoing message.

The complete modelling of the functional logic corresponding to a new system feature requires connections to be established between the analysis based on the sequence and activity diagrams corresponding to different use cases of a system, i.e. there is a need to capture the way in which messages of a sequence diagram for a particular use case of the system affects the sequence diagram of the different use case of the same system. For example, Vasilache (2015) introduced a dependency diagram to interconnect scenarios in sequence diagrams of a system. However, this dependency diagram is limited to the establishment of interconnections at a high level, and do not capture the detail of the parameters involved, which, ideally, should be annotated on the sequence diagram itself.

As a general reflection, SysML behaviour diagrams support the introduction of a formalism, but this requires the inclusion of a large amount of modelling elements with appropriate formal types (e.g. interconnections, notes, alternatives, etc.). This can make them complex and, therefore, difficult and time consuming to read even for the experienced modellers (Berkenkötter 2003; Eisenbart et al. 2015).

3 Enhanced sequence diagram notation

This section introduces the proposed concept and schema for the Enhanced Sequence Diagram (ESD), to enhance the conventional sequence diagrams with the ability to capture flow-based information associated with physical systems behaviour. The integration of flow-based functional requirements reasoning and schema with the system operational analysis based on a sequence diagram enables the ESD to provide a better support for early function and architecture analysis of complex multidisciplinary systems.

Section 3.1 introduces definitions and conventions for the representation of the elements of the ESD. Sections 3.2 and 3.3 expand on the representation of conversion

(transformation) and transmission (transaction) operations in the ESD, respectively, and Sect. 3.4 expands on the use of the ESD in the representation of branching and joining flows.

3.1 Enhanced sequence diagram schemata

In a conventional sequence diagram (SD), a “lifeline” is used to represent the lifecycle of an actor, with the implicit assumption that this revolves around operations on a single dominant flow, nominally of information. However, most engineered systems achieve their functions through concurrent or sequential operations on multiple flows, typically of material, energy or information/signal (Pahl and Beitz 2007). The fundamental idea underpinning the ESD is to add the detail of flow types and operations to a standard SD. This can be achieved by assuming that an SD lifeline can, in fact, be considered to represent a “flowline”, and that an actor could have multiple flowlines. To address this, we propose a representation schema that allows different types of operations on flows (commonly referred to as heuristics) to be represented within the ESD framework.

The fundamental representation principle for the ESD is illustrated in Fig. 3. The System of Interest (SoI) is associated with multiple flowlines to represent the flows of material (M), energy (E) and information/signal (I) through the system. Coherent with the UML/SysML sequence diagrams, the flowlines are represented by vertical lines descending from the base of the actors and the System of Interest (SoI), denoted by rectangles. Different line types are used to differentiate between flow types, using the standard Pahl and Beitz (2007) representation convention, i.e. dotted line for the flow of information/signal, thinner continuous line for energy and thicker continuous line for material as shown in Fig. 3.

Figure 3 illustrates the SoI with 3 generic flowlines for Material (M), Energy (E) and Information (I), with the Input and the Output actors represented with Energy flowlines. The SoI is treated as a black box in Fig. 3 by not including any internal actor. Internal actors can be conceptualized as subsystems or components of the SoI as the analysis of the operations sequence is captured. This makes it possible to associate multiple flowlines with the SoI to deliver the required operations on flows.

The scope line in Fig. 3 denotes the boundaries of the system use case represented in the diagram. Scope lines are used for the purpose of linking between ESDs corresponding to different use cases of a system. The initial and/or the final value(s) of the attribute(s) of the input and the output actors can be shown as states at the boundary of the scope lines, to facilitate the connection with other ESDs/use cases.

There are typically two basic operations on flows: conversion (transformation) and transmission (transaction). The former addresses the change in the attributes of an object

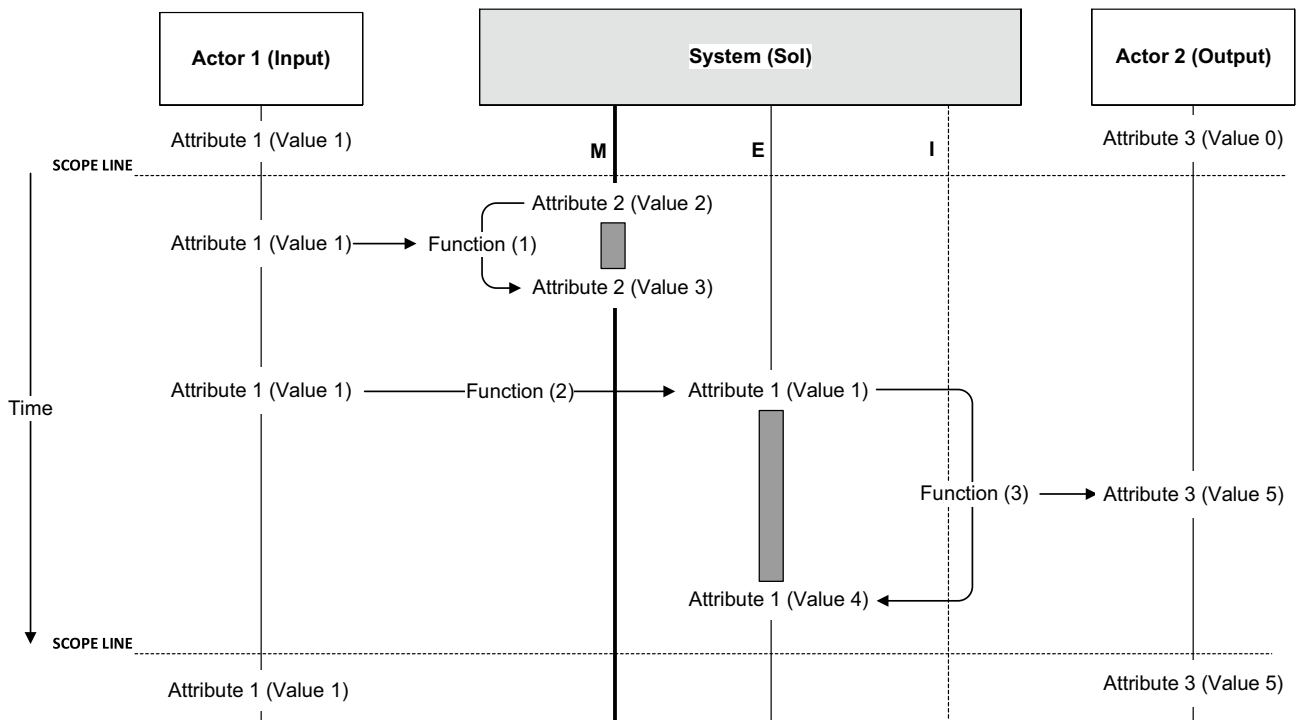


Fig. 3 Enhanced Sequence Diagram schema

that the function is applied to, while the latter is about changing the location of an object through the applied function. Achievement of function for some technical systems requires the flows to be split or joined and allocated to different functions. Development of complex functional models that combine multiple types of operations on flows is commonly underpinned by process heuristics (Maier and Rechtin 2000), to mitigate the effect of subjective judgement of the practitioner on function modeling by providing prescriptive guidelines. Our proposed approach for the ESD representation of operations on flows is underpinned by the Systems State Flow Diagram (SSFD) function modelling as described in Yildirim et al (2017). We have chosen to underpin the ESD on the SSFD functional modelling paradigm (Yildirim et al. 2017), both because of the logical complementarity, and also because this methodology has been taught and used by many of our industrial collaborators for over 10 years (Campean et al. 2011, 2013, 2015; Henshall et al. 2014; Dobryden et al. 2017; Naidu et al. 2017).

Figure 3 illustrates the ESD representation of three different types of flow operations:

- (i) Conversion (transformation) operation, illustrated by Function (1);
- (ii) Transmission (transaction) operation, illustrated by Function (2);

- (iii) Bifurcating operations on flows, illustrated by Function (3).

Jointly, these three types of flow operations define a set of heuristics for function modelling based on the ESD, which is similar to the SSFD heuristics introduced by Yildirim et al (2017). The details of the representation of the operations heuristics in the ESD schema are presented in the following sections.

3.2 ESD representation of transformative functions

The SSFD function model defines a transformative function in terms of a “triad” as shown in Fig. 4a, i.e. the SoI, conceptualized as an object described by its attributes with their input values, must interact with another object (or “actor”, denoted as “Object 1” in Fig. 4a), with certain properties (attributes/values), to achieve the desired transition to the output state, described in terms of the desired attribute (output) values of the operand. The dashed arrow from “object 1 (actor 1)” pointed to the text describes the function necessary for the transformation from the input state to the output state. A function in the SSFD is denoted by an open arrow and articulated in verb-noun format which is related by the rule that the verb corresponds to the operation on the object attribute(s) and the noun to the object or the object attribute. The corresponding ESD representation is shown in Fig. 4b,

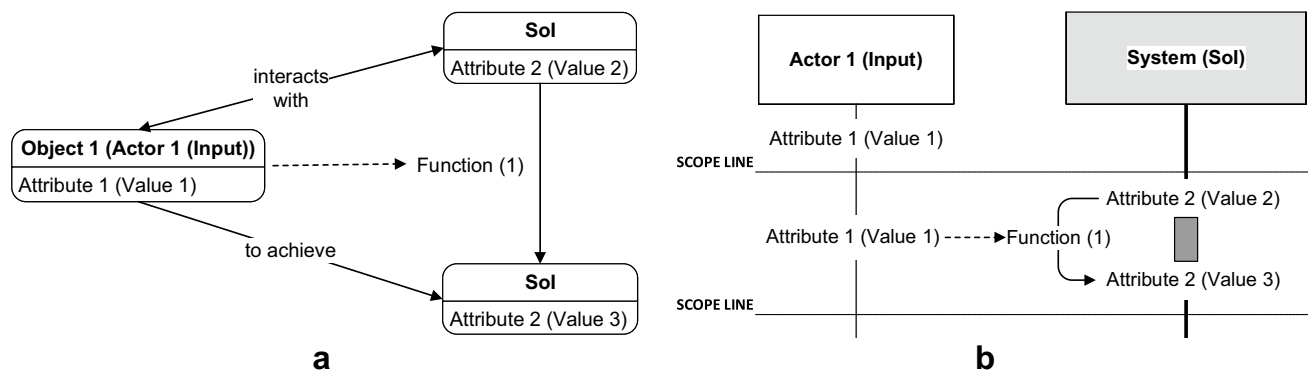


Fig. 4 Representation of a transformative function in the SSFD (a) and the ESD (b)

an excerpt from Fig. 3, with the objects in the SSFD denoted as actors in the ESD. Similarly, attributes of the objects on the SSFD are shown as the attributes of the actors on the ESD and a function is indicated with an arrow, which in the ESD links the input state to the output state.

The time over which the transformative function acts to achieve the transformation is captured in the flowline sequence in Fig. 4b, and is shown by a bar, the length of which is indicative of duration, in line with the principles of a conventional SD. Coherent with the SSFD, the function reasoning states that the transition from the SoI input state attribute 2 (value 2) to the output state attribute 2 (value 3) is achieved through interaction with the external Actor 1 attribute 1 (value 1), applied to the input state. For example, Fig. 4b shows that the SoI should import Energy from Actor 1 (Input)/Attribute 1 (Value 1), to change its Material Flow Attribute 2 from (value 2) to (value 3). A typical articulation of the functional requirement for this case, based on the flows represented in Fig. 4b, would be: “convert SoI Material Flow of Attribute 2 from value 2 to value 3”.

3.3 ESD representation of transmission functions

Function (2) in Fig. 3 illustrates the representation of a transmission function from Actor 1 (Input) to the SoI, as a straight open arrow. The assumption is that attribute 1 (value 1) is transmitted from Actor 1 to the SoI. This attribute can be associated with an internal actor in the SoI as the sequence of operations is captured.

In general, in the context of an SD, the duration of a transmission event is assumed to be negligible, as it is often the case with information systems. While it is convenient to show the representation of transmission functions in an SD, care should be taken that with physical systems (i.e. based on material and energy flows) whose transmission functions are not always synchronous and, therefore, a representation of this type of transmission functions as transformation functions (as shown in Fig. 4b) would be more appropriate.

3.4 ESD representation of branching/joining flows

Function (3) in Fig. 3 illustrates the ESD representation of a branching flow, in this case the flow of energy from the SoI to the Output Actor 2. This is illustrated in Fig. 3 as a transformative function, as the SoI Energy flow changes the value of Attribute 1 (from value 1 to value 4), changing (in this case assumed as a transmission) the value of the Attribute 3 corresponding to Actor 2 (Output) to Value 5.

For the more general case of combinations of multiple flows, the ESD notation is based on the join and fork nodes of the activity and the state machine diagrams, as illustrated in Fig. 5. Figure 5a shows a flow joining operation with a join node for the same type of flow (E) in the transition from two external actors distinguished by their attributes and attribute values (inputs) to the SoI (output). Given that the operation to combine two or more inputs towards an output requires a transformative operation (e.g. cumulating energy from 2 sources, as shown in Fig. 5a), a design element is needed to achieve this; therefore, an ESD join node is shown associated with a flowline, in this case of the same type (E) as the flows joined. The same representation principles apply to other operations on the flows of material, energy and information. To give an illustrative practical example, we can assume that Fig. 5a denotes a case where electrical signals from 2 inputs need to be evaluated to calculate a required actuation signal transmitted to the output; this could be achieved by an electronic control unit (ECU) as a possible design element associated with the flowline of the join node.

Conversely, Fig. 5b illustrates the use of a fork node in the ESD based on the same type of flow (E), showing the conversion from the SoI (input) to two external actors distinguished by their attributes and attribute values involved (outputs). An illustrative example of this case is a hydraulic brake system for a vehicle distributing the fluid energy from a master cylinder to more than one wheel to simultaneously generate a retardation force for the vehicle. Different transformations can be associated with the operation, for example

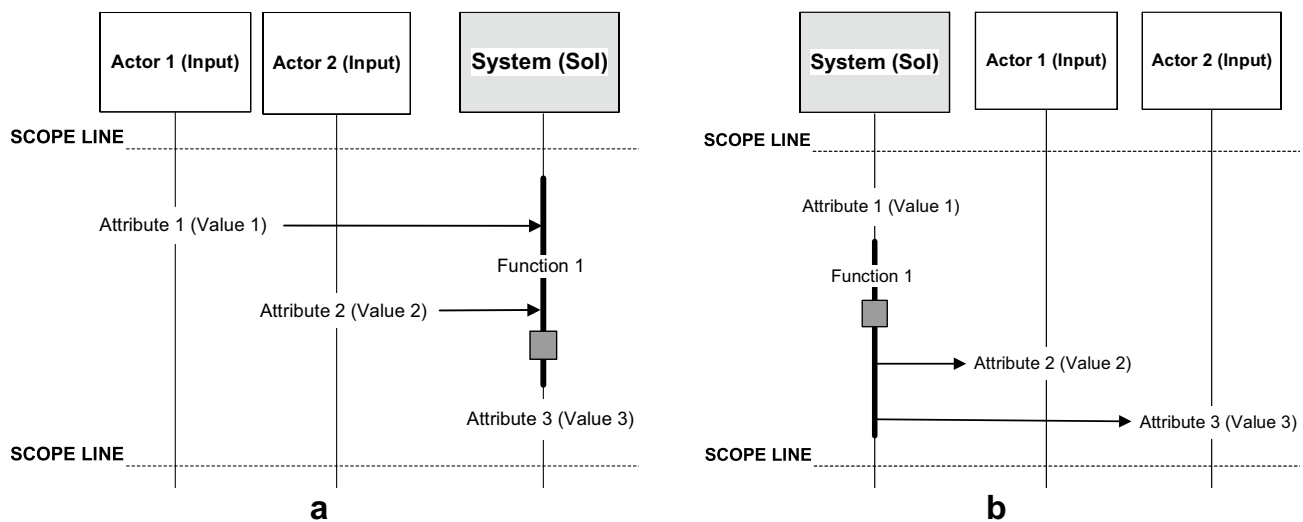


Fig. 5 ESD join node (a) and fork node (b)

in the case of an automotive engine, the combustion energy is converted to torque at the crankshaft and heat transmitted to the engine block. The same representation principle can apply to the flows of material (e.g. removal of particles from the exhaust gas flow in an aftertreatment system of a vehicle) and information (e.g. an input signal triggering an additional signal to indicate the status of a device, while the original signal input is being transmitted).

For a complex system, there will be multiple interactions between the external and internal actors, as well as interactions between internal actors, all of which need to be captured in terms of operations on flows. The ESD methodology facilitates the mapping of the sequence of exchanges (as operations on flows) between the actors towards a synthesis as a functional chain required for the achievement of the system use case. The following section provides a comprehensive illustration of the deployment of the methodology with three product development case study scenarios.

4 Case study: ESD application to the system design analysis of an electric hybrid bicycle

An Electric Hybrid Bicycle (which will be referred to as “e-bike” henceforth) was presented by Campean et al. (2017) as a research and teaching case study, to illustrate the comprehensive application of structured function and function failure analysis methodology within a complex engineering system analysis process. Several product development use case scenarios were considered to cover reverse engineering, the introduction of a new technology and introduction of new control features for customer centered innovation.

In this paper, we adopt the basis of this background case study to illustrate and validate the deployment of the ESD across a range of relevant product development use case scenarios. The following section introduces the case study methodology, outlining several modes of operation of the e-bike as use cases, including “Pedal bike”, “Pedal bike with power assistance” and “Maintain constant speed”, which are then analysed (in Sects. 4.2–4.4) in conjunction with different Product Development scenarios, to reflect the diversity of contexts faced by systems engineering designers.

4.1 Case study methodology

The development of electric bicycles (e-bikes) has taken off in recent years, primarily fuelled by the drive to offer a viable personal mobility solution for congested traffic in cities. Many automotive OEMs have also started to develop e-bikes (Manthey 2019), seen as complementary solutions to an integrated multimodal transportation system. The competition for developing advanced features for e-bikes to enhance the rider safety and experience has quickly become intense, similar in many ways with developments in automotive vehicles. For example, the e-bike of TU Delft & Gazelle includes an intelligent control assistant which intervenes when the e-bike threatens to tip over and keeps the e-bike stable and upright (Randall 2019). Blankenau et al. (2018) also described the integration of a LIDAR system to the rear axle of an e-bike, to monitor traffic situations (e.g. the identification of an oncoming vehicle).

From a user perspective, an e-bike can be driven by pedalling only (just like a conventional bicycle), and additional power can be provided by the system to complement the rider’s pedalling in different scenarios of usage. The

use case diagram in Fig. 6 illustrates several uses of the e-bike to achieve desired user goals. Common functionality shared by use cases is shown by use case relationships “inclusion” and “extension”. The inclusion relationship shows that the included use case is always fulfilled as part of the base use case; e.g. “accelerate” and “decelerate” use cases are included by “ride bike” and “maintain constant speed” use cases. The extension relationship is used to show a functionality that is generally performed by exception in respect of the base use case and it does not usually support the functionality of the base case directly. For example, “charge bike” use case in Fig. 6 extends “brake” and “motoring” use cases.

Three specific use cases of the e-bike will be considered for the purpose of the analysis in this paper, associated with different product design and development scenarios:

- (i) “Pedal bike”—illustrating the capture of generic functional analysis of a system based on reverse engineering, i.e. where the architecture is known or assumed; this is described in Sect. 4.2;
- (ii) “Pedal bike with power assistance”—illustrating a product development scenario based on the introduction of a specific new technology (electric propulsion system based on a brushless DC motor) on a conventional system; this is outlined in Sect. 4.3;

- (iii) “Cruise control”—illustrating the introduction of a smart control feature (as an advanced user assistance system) to an existing system, analysed as a logical design level, without reference to any solution; this will be discussed in Sect. 4.4.

4.2 Scenario 1: ESD analysis for “Pedal bike” use case

Figure 7 illustrates the engineering analysis scenario for the e-bike Pedal Drive System (PDS) leading to the ESD, in a structured sequence including:

- (a) The Use Case Diagram;
- (b) A black-box analysis of the PDS system using the SSFD function representation for the highlighted “Pedal bike” use case (with a clear articulation of input/output states and transformative function);
- (c) The function decomposition analysis using the ESD schema, based on the assumption of a standard architecture of a PDS including 4 subsystems: Pedals, Crank, Chain and Cogset.

The ESD diagram includes separate flowlines for each of these subsystems. Given that this system (as indicated by the SSFD Black Box analysis) is handling energy

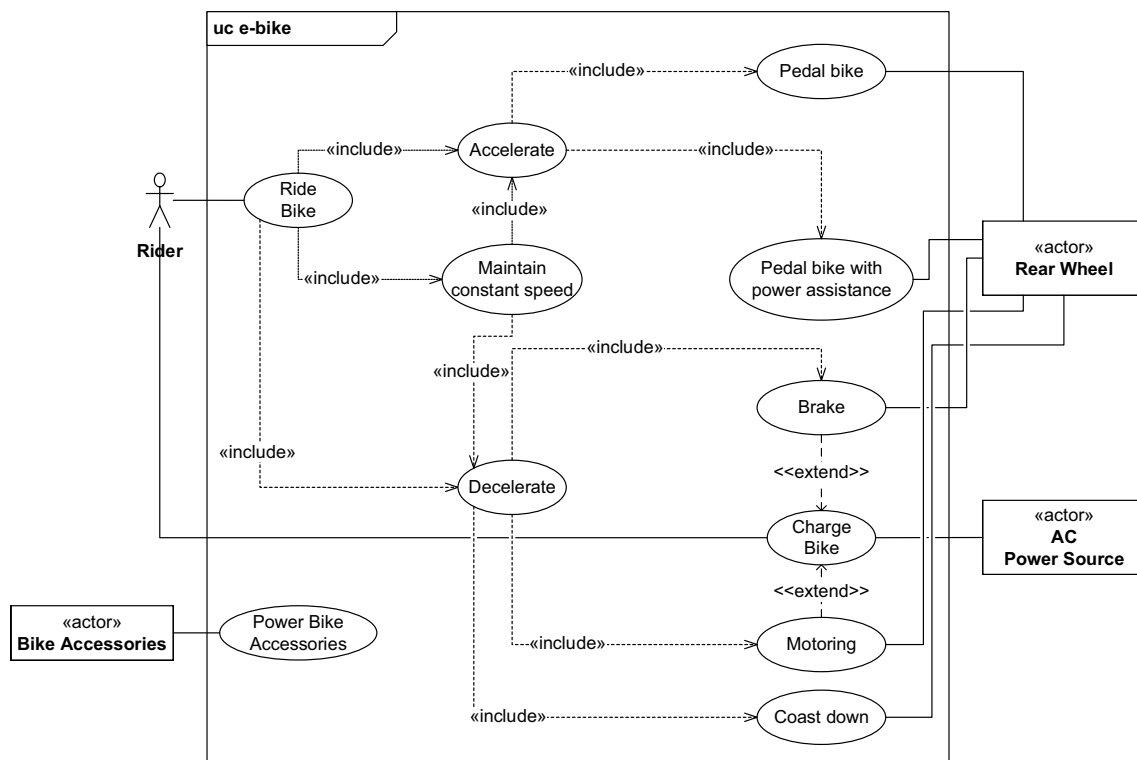


Fig. 6 Use Case Diagram for the e-bike

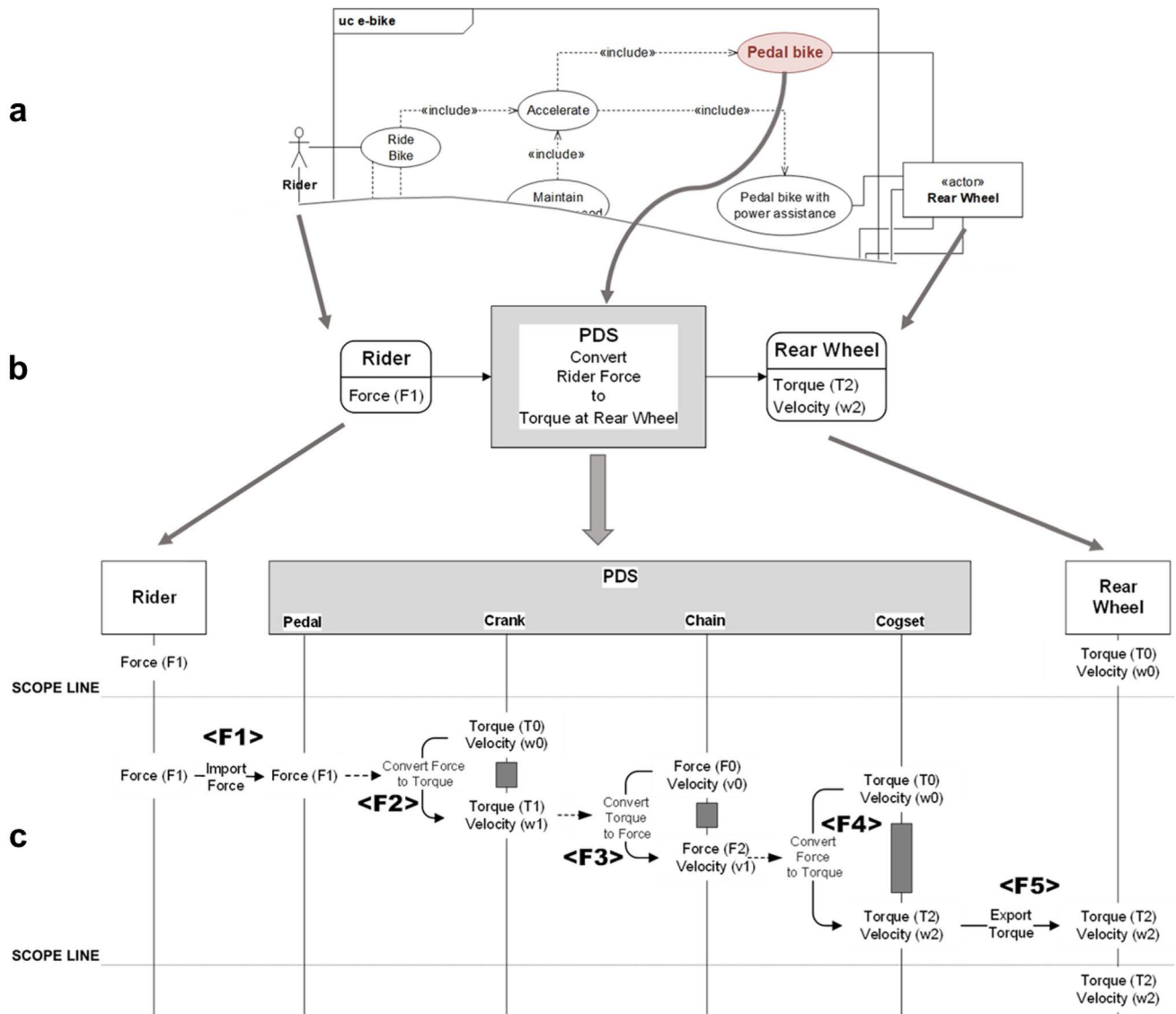


Fig. 7 ESD analysis for “Pedal bike” use case

transfer and transformation from the rider (input) to the rear wheel (output), all these flowlines are of energy type. The function flow to achieve the use case function “Convert Rider Force to Torque at Rear Wheel” is identified and represented in the ESD diagram, and includes the following functions:

- < F1> Import Force: This is a transmission function for the Pedal subsystem to enable the transmission of Rider energy, with the attribute Force (F1), to the e-bike PDS (SoI);
- <F2> Convert Force to (Rotational) Torque: This is an energy transformation (conversion) function achieved by the Crank mechanism, using the Pedal energy with the

attribute Force (F1) to increase the Torque / (Angular) Velocity from (T0, w0) to (T1, w1);

- <F3> Convert Torque to (Translational) Force: Torque (T1, w1) from the Crank is converted to Translational Force (and associated linear velocity), changing the Chain attribute values from (F0, v0) to (F2, v1);
- <F4> Convert (Translational) Force to (Rotational) Torque: The Chain attributes (F2, v1) change the Cogset attributes increasing Torque from (T0, w0) to (T2, w2);
- <F5> Export Torque: This is a transmission function (assumed lossless) of the Cogset attributes (T2, w2) to the Rear Wheel (external actor for the e-bike PDS).

4.3 Scenario 2: ESD analysis for “Pedal bike with power assistance” use case

This use case scenario requires the system to deliver to the rider’s request for a level of propulsion assistance proportional to the rider input. For example, if the rider’s instruction is a level of assistance of 30%, the e-bike will provide 30% additional torque assistance over the rider’s input converted to rotational torque (T_2 , w_2) at Cogset as shown in Fig. 7. The technology introduced to delivery this functionality includes an electric drive system (e.g. based on a brushless DC electric motor) that converts electrical energy from a battery storage to torque and a hub gear mechanism (typically a planetary gear system) to combine the mechanical torque from the PDS with the torque from the Electric Drive System (EDS). The e-bike Hybrid Propulsion System (HPS) also requires control functionality to ensure that the EDS propulsion assist delivers to the rider intent. From a Product Development scenario point

of view, we assume that the EDS technology solution has been selected, therefore, the task is to analyse and complete the design of the HPS system, including the control logic, for which the design solutions are not defined yet—so a solution agnostic analysis is required.

The ESD for this use case is illustrated in Fig. 8, emphasizing the flowlines and function decomposition associated with the hybrid functionality. The PDS functionality is abstracted to a single energy flowline as a black-box summary of its function.

The “rider” is represented as an external actor with one energy and one information flowline to capture the “intent” in a generic, solution agnostic manner. The functional logic and functional requirements of the “power assistance” technology at this level of analysis can be extracted from this diagram as follows:

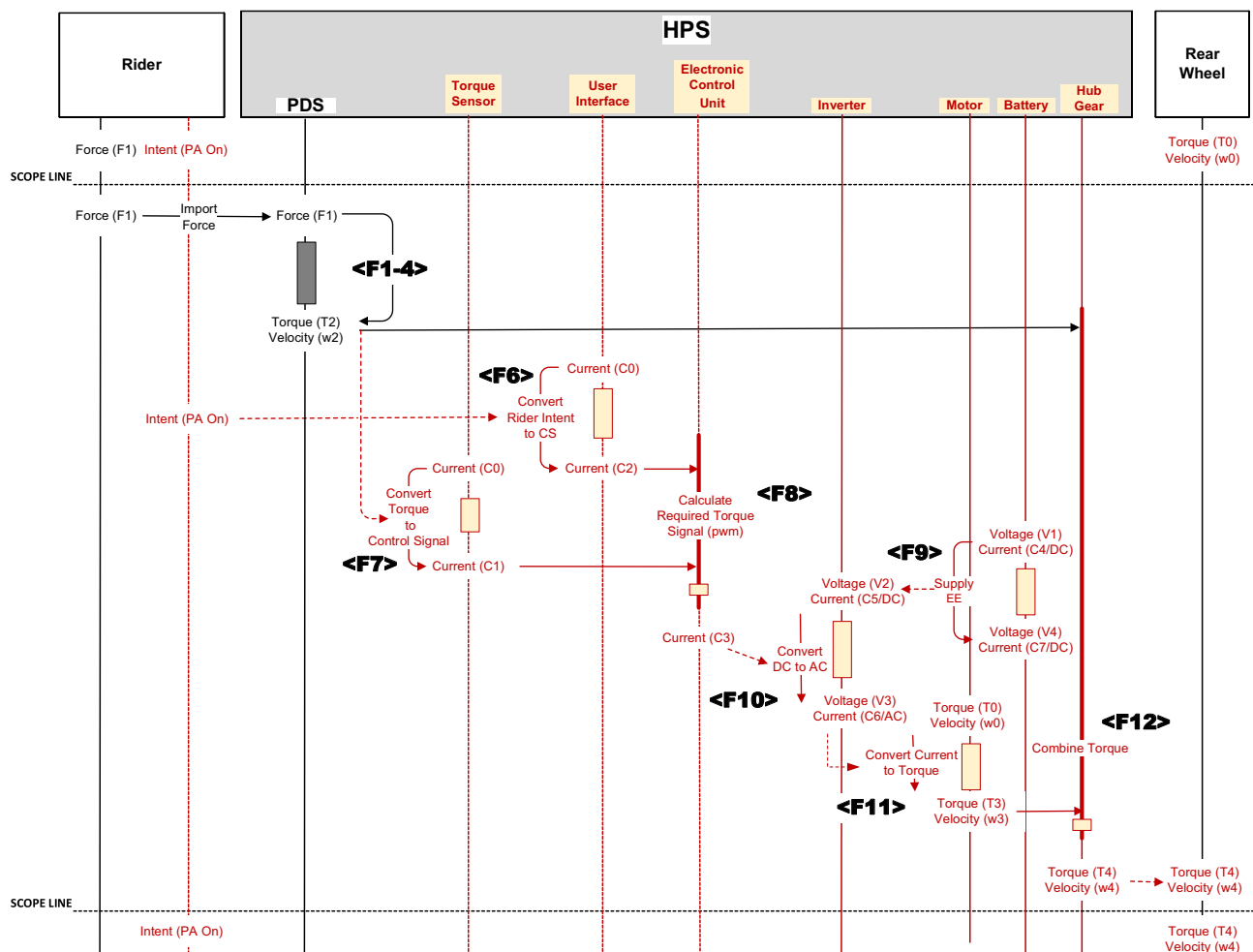


Fig. 8 ESD for “pedal bike with power assistance” use case

- <F6> Convert Rider Intent to Control Signal: Rider's Instruction (Intent—PA On) changes the attribute values of User Interface from C0 to C2;
- <F7> Convert (Rotational) Torque to Control Signal (CS): This is a conversion function; Cogset/PDS output attributes (T2, w2) change the attribute values of Torque Sensor from C0 to C1;
- <F8> Calculate Required Torque Signal: This is a conversion function whose design element Electronic Control Unit (ECU) introduced with a join node requires inputs from the Torque Sensor (C1) and the User Interface (C2) for the fulfillment of the "Calculate Required Torque Signal (C3)". Representation of C1 and C2 along with the timeline shows that C2 is received first; the ECU needs both for this function to be achieved;
- <F9> Supply Electrical Energy (EE): This is a conversion function as Battery (V1, C4/DC) provides EE for Inverter (V2, C5/DC) via a branching flow. The status of the Battery (Voltage, Current) following the fulfillment of this function can be shown at the boundary of the scope lines if required;

- <F10> Convert DC to AC: Control Signal (C3) from the ECU changes attribute values of the Inverter from (V2, C5/DC) to (V3, C6/AC);
- <F11> Convert Current to (Rotational) Torque: (V3, C6/AC) of the Inverter changes attribute values of Motor from (T0, w0) to (T3, w3);
- <F12> Combine (Rotational) Torque: In this conversion function, the Hub Gear, associated with a join node, combines Torque from the Cogset/PDS (T2, w2) with Torque of the Motor (T3, w3). The combined Torque (T4, w4) changes attribute values of the Rear Wheel from (T0, w0) to (T4, w4). As the join node requires both inputs for the generation of the output, power assistance technology will not work unless the Rider pedals the e-bike.

The allocation of multiple flowlines and relevant functions to components within the SoI enables the ESD to provide a basis for the identification of architecture on a functional basis, as shown in Fig. 9. Note that the Control sub-system, while mapped in terms of function and architecture integration, is solution agnostic—as no specific sensor

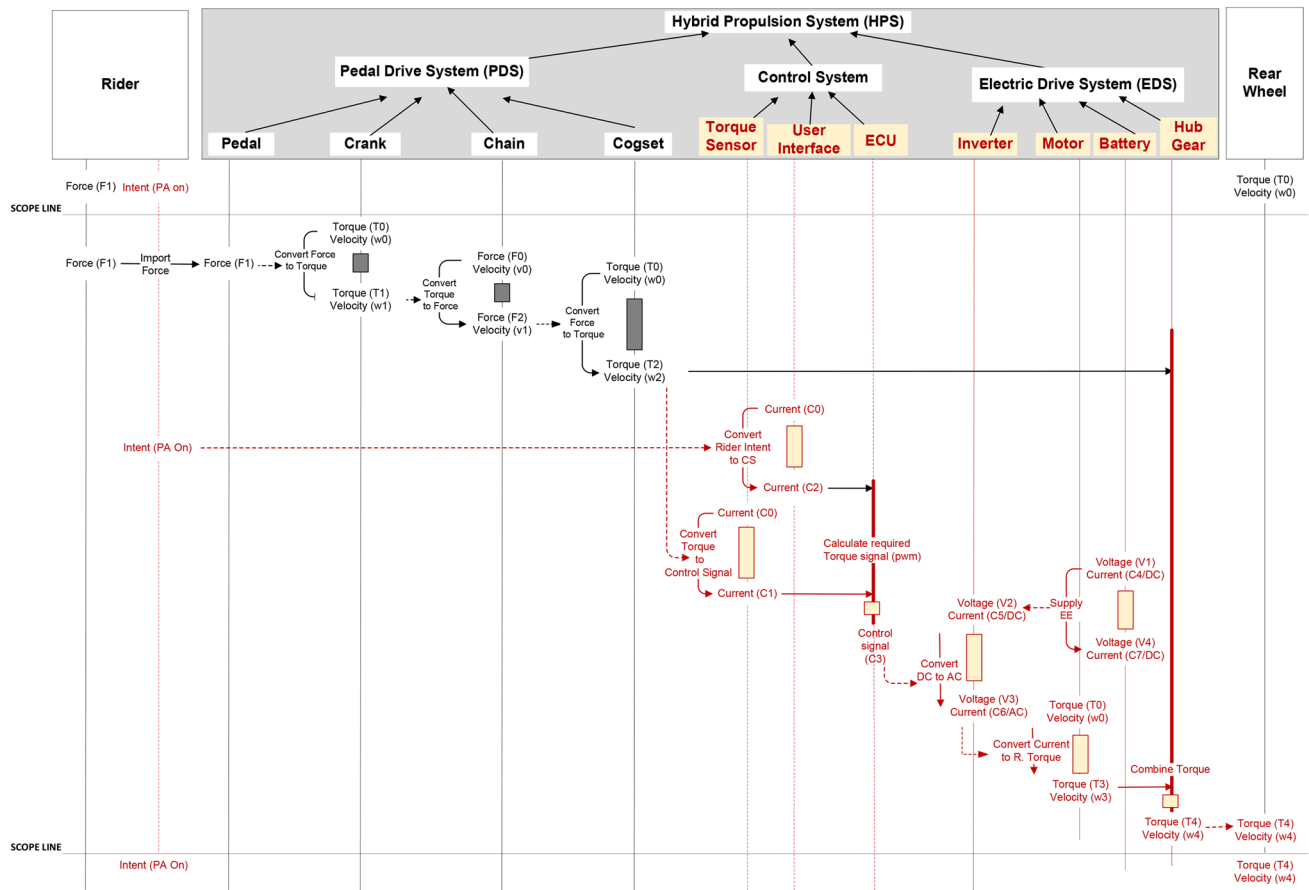


Fig. 9 Architecture development through the ESD

has yet been selected. Therefore, this level of analysis can be considered to correspond to a hybrid gray-box system modelling.

4.4 Scenario 3: ESD analysis for “Maintain constant speed” use case

“Maintain constant speed” use case in Fig. 6 can be associated with a Cruise Control (CC) feature where the rider maintains a constant speed of travel, regardless of the pedalling input and external environment such as road condition (Campean et al. 2017; Padmagirisan et al. 2019). This use case includes “accelerate (e-bike)” and “decelerate (e-bike)” use cases. The former takes place if the current speed of the e-bike is lower than the desired speed set by the user while the latter addresses the situation where the current speed of the e-bike is higher than the desired speed.

Considering the CC feature as the SoI, the purpose is to provide the user with enhanced assistance for controlling the system to achieve a certain objective: e.g. control the journey time by maintaining a constant speed. The user interaction with the feature can be achieved in multiple ways; for simplicity, we assume a use case scenario where the rider, upon achieving the desired velocity, will activate the CC feature with the intent to maintain the current speed. The functional requirements for the CC feature include confirmation of feature activation status to the user, and the implementation of a closed-loop control system to maintain the e-bike speed at the time of the CC activation. In relation to the e-bike architecture represented in Fig. 9, the CC feature proposes a different way of achievement—as a different functional logic, for the Control System architecture.

From a Product Development scenario point of view, the introduction of a CC feature to an e-bike with a hybrid propulsion system illustrates the effort towards user-centric innovation. The task of the system design team considering the CC feature as an SoI is to analyse the functional logic of the system, before considering ways in which the feature can be implemented in the existing system architecture, with the addition of new design elements or modification of existing elements.

Figure 10 represents an ESD for the CC feature based on this use case scenario functionality logic, without any assumption of specific solutions. The functional logic and requirements of the “Cruise Control” feature at this level of analysis can be extracted from this diagram and are discussed below.

User interface functionality Functions <F13–15> jointly deliver the required user interface functionality of accepting the user CC activation signal and as a response, confirming the activation status as a signal to the rider. These functions are allocated to a generic structure referred to as “User Interface-1”:

- <F13> Convert Rider Intent to Control Signal: Rider’s Instruction (Intent—CC On) changes attribute values of User Interface-1 from C0 to C8);
- <F14> Convert Control Signal (CS) to Status Display: C8 changes attribute values of the User Interface-1 from CC off to CC on;
- <F15> Display CC Status: This is a transmission function for the display of CC status to the rider.

Velocity control for the e-bike A generic structure referred to as “CC Speed Controller” requires the velocity of the e-bike at the time of feature activation to be sensed and stored, and a control signal to be generated for the EDS system based on the deviation sensed between the current/instantaneous e-bike velocity and the reference/stored value (corresponding to the intended velocity of travel). Given that for normal riding conditions the velocity of travel is directly proportional to the angular velocity of the rear wheel, the functional logic implemented in the ESD is based on this assumption.

- <F16> Convert Velocity to Control Signal: The Join node (control signal flowline C-2), given the control signal value (C8) from the User-Interface-1, will convert the current Rear Wheel angular velocity (w_4) to a Control Signal (C9);
- <F17> Convert Current Velocity to Control Signal: the status of C-1 is updated to value (C10) based on the current/instantaneous angular velocity of the Rear Wheel (w_5);
- <F18> Calculate Required Torque Signal (CRTS): This is a conversion function which is fulfilled by a generic structure referred to as ECU-1 with a join node. ECU-1 uses the inputs (C9 and C10) from the C-1 and C-2 to compute (implementing a closed-loop controller strategy) the Required Torque Signal (C11). Note that function <F18> is in fact similar to function <F8> in Fig. 8, but with a different strategy for calculating the output signal (hence control signal value (C11) rather than (C3) in the ESD shown in Fig. 8 for the “Pedal bike with Power Assistance” use case).

Based on the input from the Cogset/PDS (T2, w_2) and the CC Speed Controller (C11), the EDS system—abstracted in Fig. 10 as a join node that cumulates functions <F9–12>, will change the attribute values (T6, w_6) of the rear wheel, which represent the effect of the controller action on the velocity deviation from the target. The cycle of functions <F17–18> and <F9–12> (which includes the input from the PDS—hence the realisation of functions <F1–4>) represents a continuous control loop that maintain the CC behavior for the “maintain constant speed” use case.

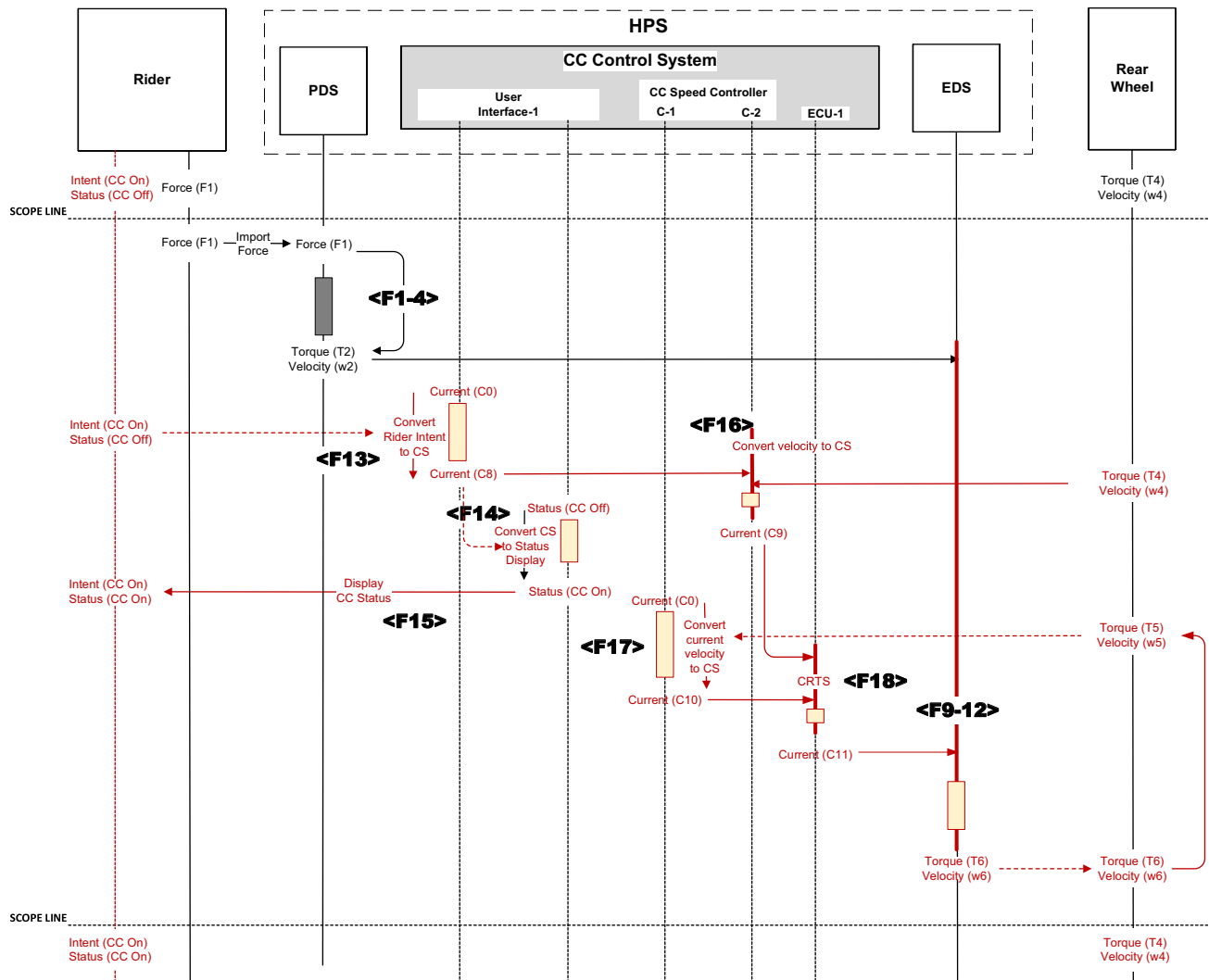


Fig. 10 ESD for “maintain constant speed” use case

5 Discussion

The following subsections reflect on the proposed ESD method in relation to the broader design and systems analysis methodologies and practice.

5.1 ESD implications for function modelling

The primary motivation for the development of the ESD method was to address the gap between the MBSE methods and tools for function analysis early in system design, and the more rigorous flow-based functional modelling basis prevalent in engineering design. While this problem has been considered by other researchers, mostly within the context of the reference frameworks of Function-Behaviour-Structure (Umeda et al. 1996; Umeda and Tomiyama 2016) and Object-Process Methodology (Dori 2016), given the rooting

of our research in industrial collaboration with automotive OEMs, we have focused on evolving a method that is commonly used and taught in the industry (the Sequence Diagram-SD), to enhance the potential impact to practice. The proposed ESD method is underpinned by a rigorous function flow modelling, integrated within the SD method commonly used for early use case analysis. The product development case study scenarios have shown that as a function modelling method the ESD can be versatile and, therefore, used as a generic function modelling methodology across the systems design analysis and development lifecycle.

While the ESD is based on the SSFD schema and modelling framework for function analysis, it brings some important insights and benefits for the functional modelling representation which have a much broader/generic value. It has been shown and discussed that the ESD representation of the operations on the flowlines and across the flowlines

maintains the time sequence characteristic of the SD, thus implicitly adding the time dimension to the function reasoning based on operations on flows. This addresses an important shortcoming of most flow-based functional modelling frameworks, that time is not captured as an explicit attribute in the functional flow representation. While preserving the SD convention for the representation of activities on a flow, the ESD explicitly captures and represents time in a graphical representation of the functional model, as illustrated in Fig. 3, as well as the e-bike case study analysis examples in Figs. 7, 8, 9 and 10. This has an important implication for design practice as it enables the representation of functional models for dynamic systems, previously recognised as being difficult (Tomiya et al. 2007). This facilitates the communication between engineers involved in the conceptual engineering and system design analysis and the engineers developing simulation models of the systems, enabling multi-disciplinary traceability of the analysis across diverse teams in Product Development.

The case study analysis presented in this paper, through the different system design scenarios, has illustrated the applicability of the ESD method to real-world complex engineering problems. In particular, the scalability of the representation to span across multiple levels of abstraction is an important feature, to enable a team to manage representations across the expanding levels of details involved with the progressive analysis of the system. For the ESD, Fig. 10 illustrates this by showing the PDS and EDS systems abstracted to a black box operation on their dominant flow—of energy. This also ensures that the mathematical correctness of the operations on flows can be verified as first-order logic models or transfer functions with the coherent mapping of parameters from the sequence diagram analysis.

5.2 ESD implications for system design and architecture analysis

The replacement of the timeline of an actor with flowlines in the ESD enables a clear mapping of transactions and transformations in terms of the measurable attributes of the actor. The inclusion of exchanges and sequences in the same diagram supports a clearer specification and articulation of the functional requirements associated with the measurable attributes that are design neutral and based on the functional basis taxonomy of Stone and Wood (2000). For example, Piques (2014) articulated a functional requirement as “Push Accelerator Pedal” in relation to an interaction between the Driver and the Hybrid Vehicle in the analysis of a hybrid vehicle and represents this requirement on the same timeline along with other requirements. This function articulation is flawed because it reflects a requirement on the driver—which is not the system of interest for the design team (it is an external

actor). It also relates to a known/assumed architecture, therefore, the functional model has limited re-use value (i.e. only for systems that inherit the particular solution referred to in the functional model). The corresponding functional requirement in the e-bike case study based on the ESD analysis was articulated as “Import Force” in Fig. 7 (functional requirement associated with the Pedal flowline). This reflects the action required on the SoI (as expected—as we engineer a system to deliver this), and promotes/enables the consideration of a variety of possible solutions. This is both highly desirable early in the product creation process, and provides a more generic, re-usable model.

This is further emphasized by the reflection that the ESD for the “Pedal bike” use case in Fig. 7 represents the allocation of the flowlines and the functions to the internal actors based on a known architecture of the e-bike. This can then be used as a “reference system” (Albert et al. 2019) in the development of new product generation through the integration of a new technology.

ESD also supports the architecture development of a system in the following ways:

- (1) The improvement of an existing product by offering alternative design choices: This can be done by considering different solutions as design elements to address functional requirements. For example, for ESD of “Pedal bike with power assistance” use case in Fig. 8:
 - (a) the user intent can be captured through a mechanical rocker switch, through a touch screen, or remotely from a mobile device;
 - (b) there could be different technical solutions for the gear hub, different electric motor technologies and different motor placements.
- (2) The improvement of an existing product for alternative scenarios: CC feature can be further developed based on different scenarios. For example, in the case of a difference between the e-bike velocity and the rear wheel angular velocity due to riding on a forest track, we can develop the analysis by adding further flowlines of information (e.g. detect slip) as required, and allocate this to design elements (e.g. slip sensor). However, this can be pursued in a more structured way with other functional and architecture development tools based on a specific architecture. For example, functional and architectural requirements of the e-bike subsystems in their external environment can be identified via an interface analysis template (Uddin et al. 2016).
- (3) Converging architectures: Multiple flowlines and functional requirements can be allocated to one internal

actor/subsystem. For example, for ESD of “Maintain constant speed” use case in Fig. 10:

- (a) C-1 and C-2 can own individual information flowlines. However, these flowlines can also be allocated to a single design element (i.e. CC Speed Controller) as shown in Fig. 10;
 - (b) Present structure (e.g. User Interface) of the e-bike in Fig. 8 can be reconsidered to include functionality (<F13–15>) of the CC feature;
 - (c) Similarly, ECU-1 could, in fact, be the same ECU referred to in Fig. 8.
- (4) The introduction of a new feature without a reference architecture: The functional logic and architecture inferred from an established abstract model provide a basis for the development of system models in industrial practice. Most common functional modelling frameworks are reasonably easy to deploy in “reverse engineering” mode. However, implementation of these functional modelling frameworks prove problematic in the introduction of new features with no current reference design to start off with. <F13–15> represents functional logic of the CC feature in terms of requirements and they could be delivered through a variety of mechanisms as solutions: from a physical switch to a touch screen (fitted on the e-bike or as remote mobile device based control), or even through gesture activated sensors.

Bottom-up architecture development in the e-bike case study, as illustrated in Fig. 9, shows that the ESD can be used at different levels of the system, i.e. from component (e.g. Torque Sensor) to system (i.e. hybrid propulsion system). The e-bike analysis could be further developed in line with other functional and architecture development tools. For example, the flows of energy between the battery and other design elements (e.g. User Interface) can be identified via an interface analysis template and relevant ESD can be updated. This process can be implemented in the opposite way for reverse engineering. For example, interface analysis template can provide a basis for the development of ESD for the e-bike “pedal bike” use case.

5.3 ESD integration with MBSE

The ESD inherits some of its features from the SysML Sequence diagram (i.e. the flowlines represented by vertical lines, bars denoting duration, and the actors and the System of Interest (SoI) denoted by rectangles) and Activity diagram (i.e. fork and join nodes). This paves the way for the formalization of the ESD schemata in SysML, however, this is beyond the scope of this paper.

The introduction of join and fork nodes in the ESD is a significant contribution to the development of conventional sequence diagrams. Unlike the ESD, current sequence diagrams represent all activities on one timeline of the SoI and actors. This leads to confusion over the differentiation of flows of material/energy/information and related activities. The ESD allows the designer to allocate multiple flowlines of energy, material and information to a single actor and a SoI. The use of different flowlines for each activity and fork/join node complement each other as the allocation of a flowline with a fork/join node clarifies what design element is responsible for the fulfilment of combining and bifurcating flows based on a sequence of activities. The use of interaction operators of sequence diagrams provides limited information in this respect. For example, Friedenthal et al. (2012) pointed out that the “Par” operator (shows which operands can occur in parallel) provides no implied order between occurrences. While “weak sequencing” introduces an order of the messages, it is challenging to visualize which messages combine or bifurcate. For example, ESD in Fig. 9 clarifies that the rider’s intent (PA on) needs to be converted to a control signal before the conversion of torque at cogset into a control signal. These activities in the ESD were associated with dedicated flowlines of information and design elements, i.e. user interface and torque sensor respectively. It is difficult to represent this on a conventional sequence diagram where the focus is rather on exchanges of messages with the representation of a single timeline for each actor and SoI.

The flowlines and associated design elements in the ESD provides a basis for the development of a SysML internal block diagram (IBD). The IBD is similar to a traditional system block diagram and shows how parts of a block are connected to each other. Figure 11 shows such a diagram for the e-bike’s CC feature extracted from the ESD analysis in Fig. 10. This conversion from ESD to IBD is straightforward as the architecture and the input/output flows are already depicted in the ESD.

Scope lines in the ESD facilitate the establishment of relationships between use cases of the e-bike. This can be done by tracing the flows of input and output attribute values in ESDs of relevant use cases. For example, the operation of the CC feature, which is represented as “maintain constant speed” use case in Fig. 6, requires the latest velocity of the e-bike from the ESD of the “pedal bike with power assistance” use case. This is shown by representing this output of the “pedal bike with power assistance” use case ESD in Fig. 8 as the input of the “maintain constant speed” use case ESD in Fig. 10, i.e. attribute values related to the latest speed of the e-bike (T4, w4). Figures 8 and 10 indicate these attributes with their values held by the Rear Wheel as output and input states at the boundary of the scope lines respectively. The same principles apply to

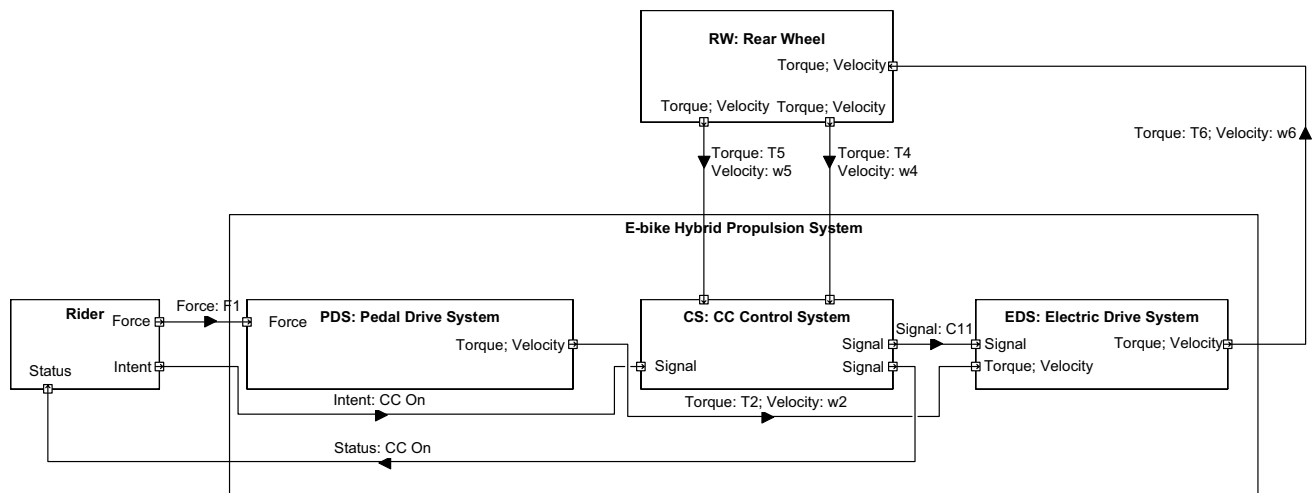


Fig. 11 E-bike CC's structural model as a SysML IBD

the internal actors as required. For example, the status of the battery can be indicated at the boundary of the scope lines as input and/or output states in Figs. 8 and 10. This can lead to the reconsideration of design decisions, e.g. existing battery may not have the capacity to cater for both “pedal bike with power assistance” and “maintain constant speed” use cases and it can be replaced with a new battery with increased capacity.

Another important reflection on the ESD graphical representation is the fact that it facilitates the transition between levels of abstraction. This is important given that most graphical functional reasoning and representation models suffer from the fact that they become increasingly complex with the expansion of the analysis towards increasing levels of detail. We have illustrated this in Fig. 8 where the PDS was represented as a single energy flowline (which is the dominant flow for the PDS) with the appropriate inputs and outputs. This enabled focus on the representation of EDS and Control System in detail—which were the subsystems being analysed, while maintaining the relevant interaction with the PDS in the overall ESD. The PDS was then expanded in Fig. 9 to provide an overall functional model representation for the hybrid propulsion system, complete at this of modelling abstraction. The EDS analysis can be further deployed to the nested functional decomposition analysis and representation of sub-systems (e.g. for the design and analysis of a User Interface that would support the integration of several user interaction features and use-cases in addition to the CC and power assistance features). In this sense, the ESD is compatible with the simulation-based MBSE modelling methods (e.g. based on Simulink or Modelica), as well as generic design frameworks based on nested systems (Crilly 2013 and 2015), or process design frameworks such as PROVE (Shaked and Reich 2019).

From the point of view of the graphical representation itself, all the e-bike analysis ESDs were completed using Microsoft Visio, with composite graphics created for the different types of functions discussed in the paper—corresponding to the flow heuristics. We recognise that further development can be done to further automate the graphical representation of the ESD, including compatibility with other MBSE/SysML packages.

5.4 ESD integration within engineering systems analysis practice

In terms of implementation of the ESD in the systems engineering design practice, i.e. in the context of other methods and methodologies that support early engineering design analysis, our reflection, based on the observations from the embedded research collaboration with a major automotive OEM, is that the greatest efficiency gain would stem from integrating the ESD with other methods. For example, we envisage that the ESD will be integrated with use case scenario analysis, with the graphical development of the ESD concurrently with the articulation of the use case scenarios—thus providing a mechanism for mutual integrity check of the logic as well as the documented analysis. We also envisage that functional requirements, extracted from the ESD, will be articulated in the appropriate systems requirements databases, which will guide the subsequent system development. The interface analysis template described by Uddin et al (2016), further enhanced and customised for use in practice as described by Campean et al. (2017), provides a powerful approach to capture the requirements based on interface reasoning—powered by the ESD analysis. The engineering system analysis can be further enhanced by function failure reasoning based on the ESD and the interface analysis template, leading to an

FMEA type analysis at the logical system level (as illustrated by Campean et al. 2017), which facilitates the early planning of verification and validation associated with the functional requirements.

6 Conclusions

This paper has introduced a rigorous framework for the functional analysis of complex multidisciplinary systems based on the proposed Enhanced Sequence Diagram (ESD).

The ESD functional representation schema integrates function flow-based analysis within a sequence diagram, enabling a more detailed and rigorous analysis of functional requirements at the very early conceptual level analysis, supporting multi-disciplinary collaboration for the development of innovative user-centred features. The ESD also offers a powerful functional analysis method for the analysis of the dynamic behaviour of complex physical systems by facilitating the coherent representation and capture of function sequences, and the explicit representation of time in a graphical functional model.

The theoretical and preliminary empirical validity (Pedersen et al. 2000) of the ESD was provided through the e-bike case study scenarios considered in the paper. This also illustrated the deployment of the ESD in different engineering analysis contexts (from reverse engineering to technology integration and new design/feature innovation) and product development scenarios. In addition to proving the completeness of the ESD schema, rooted in the SSFD flow heuristics, to represent a comprehensive diversity of flow-based operations in different scenarios and across multiple disciplinary systems, several important benefits have also been illustrated, including:

- Better functional requirements capture and articulation based on a sequence diagram analysis, deriving from the integration with well-established flow-based taxonomy of Stone and Wood (2000);
- Coupling of sequences of use cases in coherent functional models representations, with the coherent alignment of attributes and values;
- Architecture analysis integrated with graphical functional modelling and representation;
- Development of generic functional models and architecture that could easily evolve and converge with different design embodiment choices;
- Scalable and nested graphical model representations across levels of abstraction.

While the ESD research arose from the need identified through an industrial research collaboration, the rigorous

functional analysis methodology developed will be of interest and have an impact upon both the engineering design and systems engineering communities, as well as the design practice in the industry.

Acknowledgements The authors acknowledge the insight into the current industry practice of systems engineering design and analysis, gained through the engagement with the Jaguar Land Rover Technical Accreditation Scheme. The views presented in this paper are those of the authors and do not represent those of the aforementioned company. Special thanks are due to Dr Ed Henshall, Royal Academy of Engineering Visiting Professor in Integrated Systems Design who has provided insightful input for the development of the method, as well as the Editor and the anonymous referees who have generously provided valuable suggestions for improvement.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Albers A, Zingel C (2013) Challenges of model-based systems engineering: a study towards unified term understanding and the state of usage of SysML. In: Proceedings of Smart product engineering: the 23th CIRP design conference. Bochum, Germany. pp 83–92. https://doi.org/10.1007/978-3-642-30817-8_9
- Albers A, Rapp S, Spadinger M, Richter T, Birk C, Marthaler F, Heimicke J, Kurtz V, Wessels H (2019) The reference system in the model of PGE: proposing a generalized description of reference products and their interrelations. *Proc Design Soc Int Conf Eng Design* 1(1):1693–1702. <https://doi.org/10.1017/dsi.2019.175>
- Apostolov H, Fischer M, Olivotti D, Dreyer S, Breitner MH, Eigner M (2018) Modeling framework for integrated model-based development of product-service systems. *Procedia CIRP* 73:9–14. <https://doi.org/10.1016/j.procir.2018.03.307>
- Barbieri G, Fantuzzi C, Borsari R (2014) A model-based design methodology for the development of mechatronic systems. *Mechatronics* 24(7):833–843. <https://doi.org/10.1016/j.mechatronics.2013.12.004>
- Berkenkötter K (2003) Using UML 2.0 in real-time development—a critical review. In: SVERTS Workshop at the UML Conference. <https://www.verimag.imag.fr/EVENTS/2003/SVERTS/>
- Blankenau I, Zolotor D, Choate M, Jorns A et al (2018) Development of a low-cost LIDAR system for bicycles. *SAE Technical Paper* 2018-01-1051. 10.4271/2018-01-1051
- Booch G, Maksimchuk RA, Engle MW, Young BJ, Conallen J, Houston KA (2007) Object-oriented analysis and design with applications, 3rd edn. Addison-Wesley, Boston
- Borky JM, Bradley TH (2019) Effective model-based systems engineering. Springer, Switzerland

- Botham J, Dhadyalla G, Powell A, Miller P et al (2017) PICAS-SOS—practical applications of automated formal methods to safety related automotive systems. SAE Technical Paper 2017-01-0063. 10.4271/2017-01-0063
- Brusa E, Calà A, Ferretto D (2018) Systems engineering and its application to industrial product development. Springer, Switzerland. <https://doi.org/10.1007/978-3-319-71837-8>
- Burcicki D (2019) Autonomous Vehicles and Generative Design. Blog. <https://blogs.mentor.com/electricalsystems/blog/2019/02/15/generative-design-autonomous-vehicle/>
- Cameron BG (2018) MBSE: the next revolution of systems engineering. MIT, presentation
- Campean F, Henshall E, Bunson D, Day AJ, McLellan R, Hartley J (2011) A structured approach for function analysis of complex automotive systems. SAE Int J Mater Manuf 4(1):1255–1267. <https://doi.org/10.4271/2011-01-1268>
- Campean I, Henshall E, Rutter B (2013) Systems engineering excellence through design: an integrated approach based on failure mode avoidance. SAE Int J Mater Manf 6(3):389–401. <https://doi.org/10.4271/2013-01-0595>
- Campean F, Yildirim U (2017) Enhanced sequence diagram for function modelling of complex systems. Procedia CIRP 60:273–278. <https://doi.org/10.1016/j.procir.2017.01.053>
- Campean F, Williams H, Uddin A, Yildirim U (2017) Robust engineering systems analysis through failure mode avoidance. Short Course Lecture Notes, University of Bradford
- Campean F, Neagu D, Doikin A, Soleimani M, Byrne T, Sherratt A (2019) Automotive IVHM: towards intelligent personalised systems healthcare. In: Proceedings of the design society: international conference on engineering design, 1(1), 857–866. 10.1017/dsi.2019.90
- Campean F, Delaux D, Sharma S, Bridges J (2020) Reliability research roadmapping workshop: implications for engineering design. In: DESIGN Conference. Cavtat, Croatia, 26–29 October. <https://doi.org/10.21278/idc.2020.0000>
- Chen Y, Tonshal B, Mitra P, Simonds C, Aldighieri P (2015) A novel approach to the design and development of an interactive learning app for automotive IVI Systems. In: ASME international design engineering technical conferences and computers and information in engineering conference, Volume 1B: 35th Computers and Information in Engineering Conference, doi:10.1115/DETC2015-47906. <https://doi.org/10.1115/1.4034414>
- Cockburn A (2000) Writing Effective Use Cases. Addison-Wesley Professional
- Crilly N (2013) Function propagation through nested systems. Des Stud 34(2):216–242. <https://doi.org/10.1016/j.destud.2012.10.003>
- Crilly N (2015) The proliferation of functions: multiple systems playing multiple roles in multiple supersystems. Artif Intell Eng Des Anal Manuf 29(1):83–92. <https://doi.org/10.1017/S0890060414000626>
- Daily J, Gamble R, Moffitt S, Raines C et al (2016) Towards a cyber assurance testbed for heavy vehicle electronic controls. SAE Int J Commer Veh. <https://doi.org/10.4271/2016-01-8142>
- D'Amelio V, Chmarra MK, Tomiyama T (2011) Early design interference detection based on qualitative physics. Res Eng Design 22:223. <https://doi.org/10.1007/s00163-011-0108-7>
- Dobryden A, Rutter B, Hartl D, Bramson E (2017) Failure mode avoidance approach for hybrid electric vehicle systems. SAE Int J Engines. <https://doi.org/10.4271/2017-01-0298>
- Dori D (2016) Model-based systems engineering with OPM and SysML. Springer, Berlin. <https://doi.org/10.1007/978-1-4939-3295-5>
- Eigner M, Muggeo C, Dickopf T, Faißt KG (2014) An approach for a model based development process of cybertronic systems. In: Proceedings of the 58th Ilmenau Scientific Colloquium. Technische Universität Ilmenau
- Eigner M, Dickopf T, Huwig C (2016) An interdisciplinary model-based design approach for developing cybertronic systems. In: DESIGN Conference, Dubrovnik, Croatia, 16–19 May, pp.1647–1656
- Eisenbart B (2014) Supporting interdisciplinary system development through integrated function modelling. PhD dissertation. University of Luxembourg
- Eisenbart B, Mandel C, Gericke K, Blessing L (2015) Integrated function modelling: comparing the IFM framework with SysML. In: Proceedings of the 20th International Conference on Engineering Design (ICED15). Milan, 27–30 Jul
- Ferrogallini M, Linke T, Schweiger U (2019) How to boost the extended enterprise approach in engineering using MBSE—a case study from the railway business. In: Bonjour E, Krob D, Palladino L, Stephan F (eds) Complex Systems Design & Management. CSD&M 2018. Springer, Cham. https://doi.org/10.1007/978-3-030-04209-7_7
- Forsberg K, Mooz H (1998) System Engineering for Faster, Cheaper, Better. Center of Systems Management. <https://doi.org/10.1002/j.2334-5837.1999.tb00258.x>
- Friedenthal S, Moore A, Steiner R (2012) A practical guide to SysML: the systems modeling language, 2nd edn. Morgan Kaufmann, Burlington. <https://doi.org/10.1016/C2013-0-14457-1>
- Gouhar T, Jaber N, Ali MN, Garimilla SK (2018) A dynamic GUI Platform for bluetooth automotive application voice communication package. SAE Technical Paper 2018-01-0023. doi:10.4271/2018-01-0023
- Graessler I, Hentze J, Bruckmann T (2018) V-models for interdisciplinary systems engineering. In: DESIGN Conference, Dubrovnik, Croatia, pp.747–756. <https://doi.org/10.21278/idc.2018.0333>
- Hehenberger P, Vogel-Heuser B, Bradley D, Eynard B, Tomiyama T, Achiche S (2016) Design, modelling, simulation and integration of cyber physical systems: methods and applications. Comput Ind 82:273–289. <https://doi.org/10.1016/j.compind.2016.05.006>
- Henshall E, Campean I, Rutter B (2014) A systems approach to the development and use of FMEA in complex automotive applications. SAE Int J Mater Manuf 7(2):280–290. <https://doi.org/10.4271/2014-01-0740>
- Henshall E, Rutter B, Souch D (2015) Extending the role of interface analysis within a systems engineering approach to the design of robust and reliable automotive product. SAE Int J Mater Manuf 8(2):322–335. <https://doi.org/10.4271/2015-01-0456>
- Holt J, Perry S (2019) SysML for systems engineering: a model-based approach, 3rd edn. The Institution of Engineering and Technology, London. <https://doi.org/10.1049/PBPC020E>
- Karkare P (2018) Integrating complex hardware and software processes for autonomous vehicle development. White Paper. <https://www.plm.automation.siemens.com/global/en/topic/autonomous-vehicles-development/52924>
- Karrer-Müller E, Fuss T, Schiffer M, Kreimeyer M, Schuh G (2016) Towards implementing systems engineering as part of commercial vehicle design. In: DESIGN Conference, Dubrovnik, Croatia, 16–19 May, pp 1669–1678
- Lindemann U, Maurer M, Braun T (2009) Structural complexity management: an approach for the field of product design. SpringerVerlag, Berlin. <https://doi.org/10.1007/978-3-540-87889-6>
- Liu C, Hildre HP, Zhang H, Rolvag T (2015) Conceptual design of multi-modal products. Res Eng Design 26:219–234. <https://doi.org/10.1007/s00163-015-0193-0>
- Manthey N (2019) Volkswagen (VWCV) presents serial cargo e-bike. electrive today. <https://www.electrive.com/2019/05/05/volkswagen-vwcv-presents-serial-cargo-e-bike/>
- Maier MW, Rechtin E (2000) The art of systems architecting, 2nd edn. CRC Press, Boca Raton
- Markos P, Dentsoras A (2015) Application of substract and operate method for developing function energy structures of products

- and systems—a rule-guided approach. In: Proceedings of the 20th international conference on engineering design (ICED15), Milan, Italy, 27.–30.07
- Micouin P (2014) Model-based systems engineering: fundamentals and methods. Wiley, USA. <https://doi.org/10.1002/9781118579435>
- Micskei Z, Waeselynck H (2011) The many meanings of UML 2 sequence diagrams: a survey. *Softw Syst Model* 10:489–514. <https://doi.org/10.1007/s10270-010-0157-9>
- Naidu A, Brittle P, Ma X, Rutter B (2017) Integrated systems engineering approach for incremental 48 volt hybrid technology introduction. SAE Technical Paper 2017-01-1603. 10.4271/2017-01-1603
- OMG (2017) OMG Unified Modeling Language (OMG UML®) Specification Version 2.5.1. <https://www.omg.org/spec/UML/2.5.1/PDF>
- OMG (2019) OMG Systems Modeling Language (OMG SysML™) Specification Version 1.6. <https://sysml.org/res/docs/specs/OMGSysML-v1.6-19-11-01.pdf>
- Padmagirisan P, Sowmya R, Sankaranarayanan V (2019) Power-assist control of a human–electric hybrid bicycle with energy regeneration and cruise control. *Proc Inst Mech Eng Part I J Syst Control Eng* 233(2):179–191. <https://doi.org/10.1177/0959651818788776>
- Pahl G, Beitz W, Feldhusen J, Grote KH (2007) Engineering design: a systematic approach, 3rd edn. Springer, London. <https://doi.org/10.1007/978-1-84628-319-2>
- Pedersen K, Emblemstvag J, Bailey R, Allen JK, Mistree F (2000) Validating design methods and research: the validation square. In: Proceedings of ASME design engineering technical conference, Paper No. ASME DETC 2000/DTM-14579, Baltimore, MD, September 10–14
- Pfeffer R, Basedow GN, Thiesen N, Spadinger M, Albers A, Sax E (2019) Automated driving—challenges for the automotive industry in product development with focus on process models and organizational structure. IEEE International Systems Conference, Florida. <https://doi.org/10.1109/SYSCON.2019.8836779>
- Piques J-D (2014) SysML for embedded automotive systems: SysCARS methodology. Embedded Real Time Software and Systems (ERTS). Toulouse, 05–07 February
- Randall C (2019) TU Delft & Gazelle developing crash-proof e-bike. *electrive today*. <https://www.electrive.com/2019/04/25/tu-delft-and-gazelle-developing-fall-proof-e-bike/>
- SDL-RT (2013) Specification and description language—real time Version 2.3. <https://www.sdl-rt.org/standard/V2.3/pdf/SDL-RT.pdf>
- Shaked A, Reich Y (2019) Designing development processes related to system of systems using a modeling framework. *Syst Eng SOSE Spec Issue* 22(6):561–575. <https://doi.org/10.1002/sys.21512>
- Short AR, Van Bossut DL (2015) Rerouting failure flow using logic blocks in functional models for improved system robustness: failure flow decision functions. In: Proceedings of the 20th international conference on engineering design (ICED15), Milan, Italy, 27.–30.07
- Stone RB, Wood KL (2000) Development of a functional basis for design. *J Mech Des* 122:359–370. <https://doi.org/10.1115/1.1289637>
- Tang J, Zhu S, Faudou R, Gauthier JM (2019) An MBSE framework to support agile functional definition of an avionics system. In: Bonjour E, Krob D, Palladino L, Stephan F (eds) *Complex Systems Design & Management*. CSD&M 2018. Springer, Cham. https://doi.org/10.1007/978-3-030-04209-7_14
- Tomiya T, D’Amelio V, Urbanic J, ElMaraghy W (2007) Complexity of multi-disciplinary design. *Ann CIRP* 56(1):185–188. <https://doi.org/10.1016/j.cirp.2007.05.044>
- Uddin A, Campean F, Khan MK (2016) Application of the interface analysis template for deriving system requirements. In: Proceedings of international design conference (DESIGN), pp. 543–552, Dubrovnik, Croatia, May 16–19
- Ullman DG (2010) The mechanical design process. 4th edn. New York: McGraw-Hill
- Ulrich KT, Eppinger SD (2003) Product design and development, 3rd edn. McGraw-Hill/Irwin, New York
- Umeda Y, Ishii M, Yoshioka M, Shimomura Y, Tomiyama T (1996) Supporting conceptual design based on the function-behavior-state modeller. *Artif Intell Eng Des Anal Manuf* 10(4):275–288. <https://doi.org/10.1017/S0890060400001621>
- Umeda Y, Tomiyama T (2016) Development of function modeling and its application to self-maintenance machine. In: Chakrabarti A, Lindemann U (eds) *Impact of design research on industrial practice*. Springer, New York. https://doi.org/10.1007/978-3-319-19449-3_20
- Weilkiens T (2006) Systems engineering with SysML/UML: modeling, analysis, design. Morgan Kaufmann, Burlington. <https://doi.org/10.1016/B978-0-12-374274-2.X0001-6>
- Xie S, Kraemer E, Stirewalt REK, Dillon LK, Fleming SD (2009) Design and evaluation of extensions to UML sequence diagrams for modelling multithreaded interactions. *Inf Vis* 8(2):120–136. <https://doi.org/10.1057%2Fivs.2009.6>
- Vasilache, S. (2015) DePAT: A Framework for Specification-Based Automatic Test Generation. In: Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS), Vol 1, 18–20 March, Hong Kong, pp. 517–522.
- Van Beek TJ, Tomiyama T (2009) Integrating conventional system views with function-behaviour-state modelling. In: Proceedings of CIRP Design Conference, Cranfield University, March 30–31
- Yildirim U (2015) Function modelling of complex multidisciplinary systems: development of a system state flow diagram methodology for function decomposition of complex multidisciplinary systems. PhD Thesis. University of Bradford
- Yildirim U, Campean F, Williams H (2017) Function modeling using the system state flow diagram. *Artif Intell Eng Des Anal Manuf* 31(4):413–435. <https://doi.org/10.1017/S0890060417000294>
- Zingel C, Albers A, Matthiesen M, Maletz M (2012) Experiences and advancements from one year of explorative application of an integrated model-based development technique using C&C²-A in SysML. *Int J Comput Sci (IJSC)* 39(2):165–181
- Zurita NFS, Stone RB, Demirel HO, Tumer IY (2018) The function-human error design method (FHEDM). Proceedings of the ASME 2018 international design engineering technical conferences & computers and information in engineering conference—IDETC/CIE 2018. August 26–29, 2018, Quebec, Canada. <https://doi.org/10.1115/DETC2018-85327>

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.